Remarks

Reconsideration of the application and allowance of all pending claims are respectfully requested. **Applicants respectfully request careful consideration of the remarks being presented herein, since they address the newly applied art.** Claims 1-22, 24 and 27-44 remain pending.

In the Office Action, dated May 19, 2005, claims 1-11, 13-22, 24, 27-39 and 41-44 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,263,489 to Olsen et al.; and claims 12 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olsen in view of U.S. Patent No. 5,819,093 to Davidson et al. Applicants respectfully, but most strenuously, traverse these rejections for the reasons herein.

Applicants' invention is directed, in one example, to automatically restoring debugging breakpoints subsequent to program code modification. The automatically restoring restores a breakpoint to the same step the breakpoint was set prior to modification of the program, although that step is now at a different location within the modified program.

In order to automatically restore the debugging breakpoint to the same step of the program that had the breakpoint prior to modification, the location (e.g., line of code) that includes that selective step is determined. This determination is made by creating an instruction profile that includes attributes of a number of generated instructions that are associated with the selected step. These attributes include attributes of the instruction (such as, operands and operation type obtained from reading the instruction) of the selected step, and possibly attributes of other instructions in proximity to the selected step. This is described further with reference to the following program:

Source View On Debugger Front End	**Machine Instructions In Application**
Line 1: /*Test program 1*/	00001 L 1,4,0
Line 2: int i = 0;	00002 L 1,5,1
Line 3: int j = 1;	00003 TC 1,4,5
Line 4: if (i > j) {	00004 BL 6
Line 5: print hello;	00005 P hello
Line 6: }	00006 TC 1,4,5

Line 7: if (j > i) {	00007 BH 9
Line 8: print world;	00008 P world
Line 9: }	00009 G back
Line 10: exit;	

In this example, a line breakpoint is set for line 7 in the Source View, which corresponds to line 6 of the Machine Instructions. However, to describe line 6, which looks like line 3, attributes of line 7 of the Machine Instructions are also included in the instruction profile in order to uniquely identify line 6 as the line in which the breakpoint is associated with. The number of instructions to be included in the instruction profile is determined based on a calibration technique described in applicants' specification.

When the program is modified, the attributes of the instruction profile are compared to the attributes of the newly generated instructions to determine where the particular line of code of interest (e.g., line 6) has moved in the program. The various comparisons made to different instructions within the modified program yield difference counters. In this example, the difference counter having the smallest value indicates the location of the selected step. Thus, the debugger automatically restores the breakpoint to that step.

In one particular example, applicants claim a method of restoring debugging breakpoints, in which the method includes, for instance, having a breakpoint that is set to a selected step of a program, the program being absent of embedded debug commands; and automatically restoring, after modification of the program, the breakpoint to the selected step, wherein the selected step is at a different location within the modified program. Thus, in this aspect of applicants' claimed invention, a debugging breakpoint is automatically restored to the same step in which the breakpoint was set before the program was modified. This is very different from the teachings of Olsen.

In Olsen, a technique is described for debugging optimized code. While in Olsen a breakpoint is set to a step in the source code, there is no description, teaching or suggestion of automatically restoring after modification of the program, the breakpoint to that same selected step, as claimed by applicants. In contrast, in Olsen, a user sets a source breakpoint, P, and therefrom watermarks are determined and used in executing the program. The watermarks include high water marks that describe steps after the breakpoint, and low water

marks that describe steps prior to the breakpoint. In particular, the high water marks describe a range of steps that are performed after the line of code with the breakpoint, and the low water marks describe a range of steps that are performed prior to that source line. During execution of the program, the high water marks and low water marks are used to determine the instructions to perform (see, e.g., Col. 12, lines 1-8).

There is no description, teaching or suggestion in Olsen of restoring the breakpoint to the same selected step that had the breakpoint set before modification. Instead, Olsen sets the breakpoint and then relies on the high water marks and low water marks to determine the flow of execution. There is no discussion of restoring the breakpoint, P, once the program is modified. This is not needed, since high water marks and low water marks are relied upon.

In the Office Action, support for the rejection of the restoring element claimed by applicants is indicated at Col. 13, lines 8-12. However, applicants respectfully submit that the cited text merely describes substituting a breakpoint instruction for the instruction at each high water mark. The high water mark is not the step for which the breakpoint was previously set. Thus, the cited text does not describe, teach or suggest restoring the breakpoint at the same step at which the breakpoint was set.

Applicants respectfully submit that Olsen does not teach automatically restoring the breakpoint to the same step, as claimed by applicants, because Olsen is focused on a different problem than that of applicants. In Olsen, the source program, itself, is not modified to a new version; it is just optimized. Therefore, the focus is on performing the instructions of the optimized code in the correct order. However, with applicants' claimed invention, the source code itself is actually modified to a new version, and hence, the focus is on finding the corresponding line of code that had the breakpoint, and automatically resetting the breakpoint to that same line, as before.

Since Olsen fails to describe, teach or suggest automatically restoring, after program modification, the breakpoint to the same selected step in which it was set prior to modification, applicants respectfully submit that Olsen does not anticipate applicants' claimed invention.

Based on the foregoing, applicants respectfully submit that independent claim 1, as well as the other independent claims, is patentable over Olsen. Further, the dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features.

Should the Examiner wish to discuss this case with applicants' attorney, please contact applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller

Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: July 19, 2005.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579